## CLAIMS

1. A method for handling queue overflow in an in-cache garbage collection process, comprising:

scanning a first object cache to identify live objects, references to the identified live objects being stored to a first broadcast queue associated with the first object cache if the identified live objects reside in the first object cache;

for live objects in the first object cache that were not processed due to an overflow of the first broadcast queue, setting bits in a register to identify portions of the first object cache that include live objects that were not processed; and

rescanning the first object cache in only the portions of the first object cache that are identified by the bits set in register.

2. A method for handling queue overflow in an in-cache garbage collection process as recited in claim 1, further comprising:

storing references to the live objects not associated with the first object cache to broadcast queues other than the first broadcast queue.

3. A method for handling queue overflow in an in-cache garbage collection process as recited in claim 2, wherein before the storing of the references to the live objects, the method comprises:

broadcasting identification of the live objects, the broadcasting enabling the first broadcast queue and other broadcast queues to determine whether a reference to one of

the live objects is to be stored in the first broadcast queue or one of the other broadcast queues.

4. A method for handling queue overflow in an in-cache garbage collection process as recited in claim 1, wherein the references to the live objects stored in the first broadcast queue are from a referencing object.

5. A method for handling queue overflow in an in-cache garbage collection process as recited in claim 1, wherein the first broadcast queue operates as first-in-first-out (FIFO) storage.

6. A method for handling queue overflow in an in-cache garbage collection process as recited in claim 1, wherein cache lines of live objects in the first object cache that were not processed due to the overflow of the first broadcast queue maintain an unset done bit.

7. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment, comprising:

an object cache having a broadcast queue;

a register associated with the object cache, each bit of the register identifying a portion of the object cache;

a controller for managing the object cache, the controller being configured such that an overflow of the broadcast queue causes the controller to set a bit in the register,

the set bit identifying the portion of the object cache that includes an object that was identified as live but not processed due to the overflow of the broadcast queue; and

a processor configured to use set bits in the register to limit rescanning to only the portions of the object cache that include live but unprocessed objects.

8. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 7, wherein the multi-processor environment includes two or more sets of (i) an object cache, (ii) a broadcast queue, and (iii) a register, such that scanning of any one of the object caches enables selected broadcast queues to store object references for objects contained in their respective object cache.

9. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 7, wherein each bit of the register is defined as an overflow partition bit.

10. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 7, wherein each object in the object cache includes an object identifier (OID).

11. A system for managing queue overflow in an in-cache garbage collection process as recited in claim 7, wherein the broadcast queue operates as a first-in-first-out (FIFO) storage.

12. A method for handling queue overflow in an in-cache garbage collection process carried out in a multi-processor environment, comprising:

scanning one or more object caches;

broadcasting references for live objects found in the one or more object caches;

storing the references in particular broadcast queues, such that references to objects found in one object cache are stored in a respective broadcast queue;

if an overflow of any one of the broadcast queues occurs, setting a bit in overflow partition bits associated with a respective object cache and broadcast queue; and

after the one or more object caches have been scanned once, rescanning only selected portions of the one or more object caches that include references that were marked but not processed, the selected portions of the one or more object caches being identified by set bits in the overflow partition bits associated with each respective object cache and broadcast queue.

13. A method for handling queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 12, further comprising:

scanning objects in the broadcast queues to find any recursive references.

14. A method for handling queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 12, wherein each bit in the overflow partition bits is associated with a respective object cache and broadcast queue and corresponds to a non-overlapping piece of the respective object cache.

15. A method for handling queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 14, wherein each non-overlapping piece of the respective object cache is of equal size.

16. A method for handling queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 12, further comprising:

before a selected portion of the one or more object caches has been rescanned, unsetting the bit in the overflow partition bits that identifies the selected portion of the one or more object caches.

17. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment, comprising:

an object cache having a broadcast queue;

overflow partition bits associated with the object cache, each bit in the overflow partition bits identifying a portion of the object cache; and

a controller for managing the object cache, the controller being configured such that an overflow of the broadcast queue causes the controller to set a bit in the overflow partition bits, the set bit identifying the portion of the object cache that includes an object that was marked but not processed due to the overflow of the broadcast queue; and

a processor configured to use the set bits to limit rescanning to only the portions of the object cache that include marked but unprocessed objects.

18. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 17, wherein the multi-processor environment includes two or more sets of (i) an object cache, (ii) a broadcast queue, and (iii) overflow partition bits, such that scanning of any one of the object caches enables selected broadcast queues to store objects contained in their respective object cache.

19. A system for managing queue overflow in an in-cache garbage collection process carried out in a multi-processor environment as recited in claim 17, wherein each object in the object cache includes an object identifier (OID).

20. A system for managing queue overflow in an in-cache garbage collection process as recited in claim 17, wherein the broadcast queue operates as a first-in-first-out (FIFO) storage.